Matplotlib

November 27, 2024

1 Use of Data Visualization

Relationships between variables Composition of the data over time Distribution of variable(s) in data Comparison of data with relation to time, variables, categories, etc.

2 matplotlib

Matplotlib is hierarchically organized. A matplotlib object is a tree-like structure of matplotlib objects which build a "hierarchy". The top of the tree-like structure of matplotlib objects is the figure object. A figure can be seen as the container which contains one or more plots. The plots are called axes in matplotlib jargon. Axes is the plural of the word "axis", which gives us a misleading idea. We can think about "axes" as a synonym for the word "plot".

3 Matplolib Objects

Figure: Outermost container for a Matplotlib graphic. Can contain multiple Axes objects.

Axes: Actual plots. Contain smaller objects (tick marks, individual lines, etc.)

Artist: Everything that is seen on the figure is an artist.

[2]:
```
pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp312-cp312-win_amd64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp312-cp312-win_amd64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.54.1-cp312-cp312-win_amd64.whl.metadata (167 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp312-cp312-win_amd64.whl.metadata (6.4 kB)
Requirement already satisfied: numpy>=1.23 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from matplotlib)
(1.26.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from matplotlib) (24.1)
Collecting pillow>=8 (from matplotlib)
```

```
  Downloading pillow-11.0.0-cp312-cp312-win_amd64.whl.metadata (9.3 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: six>=1.5 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
Downloading matplotlib-3.9.2-cp312-cp312-win_amd64.whl (7.8 MB)
   ---------------------------------------- 0.0/7.8 MB ? eta -:--:--
   - -------------------------------------- 0.3/7.8 MB ? eta -:--:--
   ------ --------------------------------- 1.3/7.8 MB 4.2 MB/s eta 0:00:02
   ---------- ----------------------------- 2.1/7.8 MB 4.3 MB/s eta 0:00:02
   ---------------- ----------------------- 3.1/7.8 MB 4.3 MB/s eta 0:00:02
   ---------------- ----------------------- 3.1/7.8 MB 4.3 MB/s eta 0:00:02
   -------------------------------- ------- 6.3/7.8 MB 6.0 MB/s eta 0:00:01
   --------------------------------------- - 7.6/7.8 MB 6.3 MB/s eta 0:00:01
   ---------------------------------------- 7.8/7.8 MB 5.6 MB/s eta 0:00:00
Downloading contourpy-1.3.0-cp312-cp312-win_amd64.whl (218 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.54.1-cp312-cp312-win_amd64.whl (2.2 MB)
   ---------------------------------------- 0.0/2.2 MB ? eta -:--:--
   --------------------------------------- - 2.1/2.2 MB 13.1 MB/s eta 0:00:01
   ---------------------------------------- 2.2/2.2 MB 10.4 MB/s eta 0:00:00
Downloading kiwisolver-1.4.7-cp312-cp312-win_amd64.whl (55 kB)
Downloading pillow-11.0.0-cp312-cp312-win_amd64.whl (2.6 MB)
   ---------------------------------------- 0.0/2.6 MB ? eta -:--:--
   ---------------------------------------- 2.6/2.6 MB 12.3 MB/s eta 0:00:00
Downloading pyparsing-3.2.0-py3-none-any.whl (106 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler,
contourpy, matplotlib
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.54.1
kiwisolver-1.4.7 matplotlib-3.9.2 pillow-11.0.0 pyparsing-3.2.0
Note: you may need to restart the kernel to use updated packages.
```

[80]:
```
pip install seaborn
```

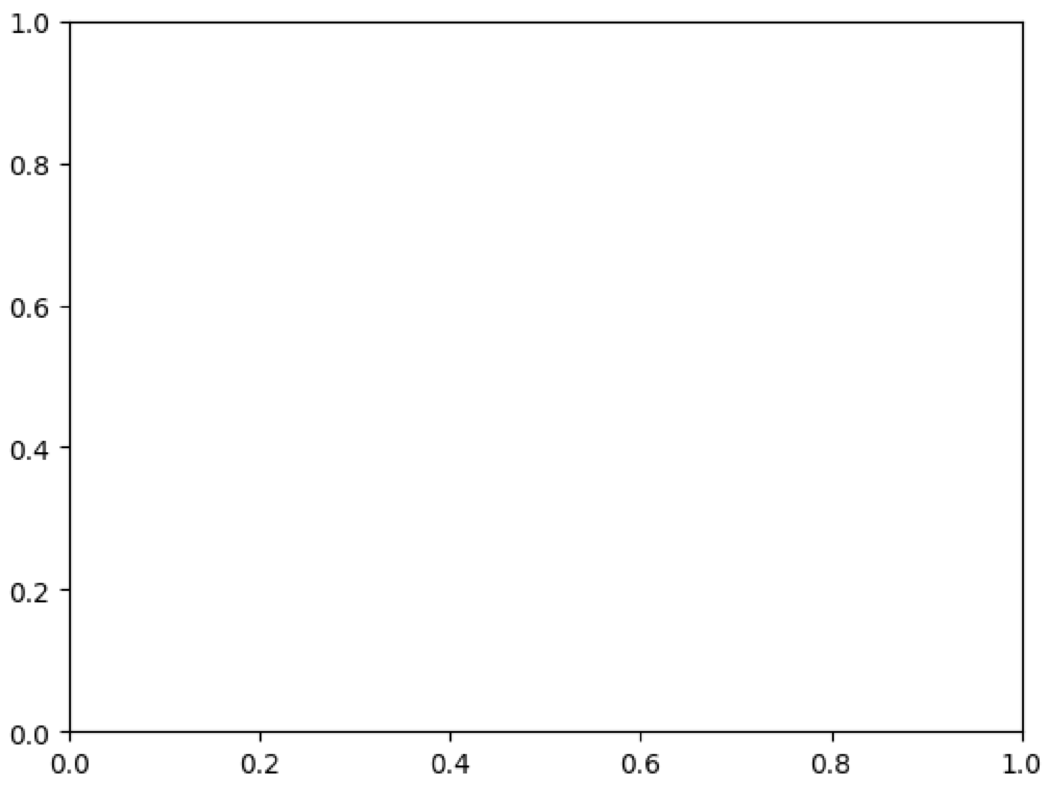
```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from
pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in
c:\users\mksin\anaconda3\envs\pandas\lib\site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Note: you may need to restart the kernel to use updated packages.
```

[4]:
```python
import matplotlib.pyplot as plt
```

3.1 GENERATE FIGURE AND AXES

[57]:
```python
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
print(type(fig))
print(type(ax))
```

```
<class 'matplotlib.figure.Figure'>
<class 'matplotlib.axes._axes.Axes'>
```



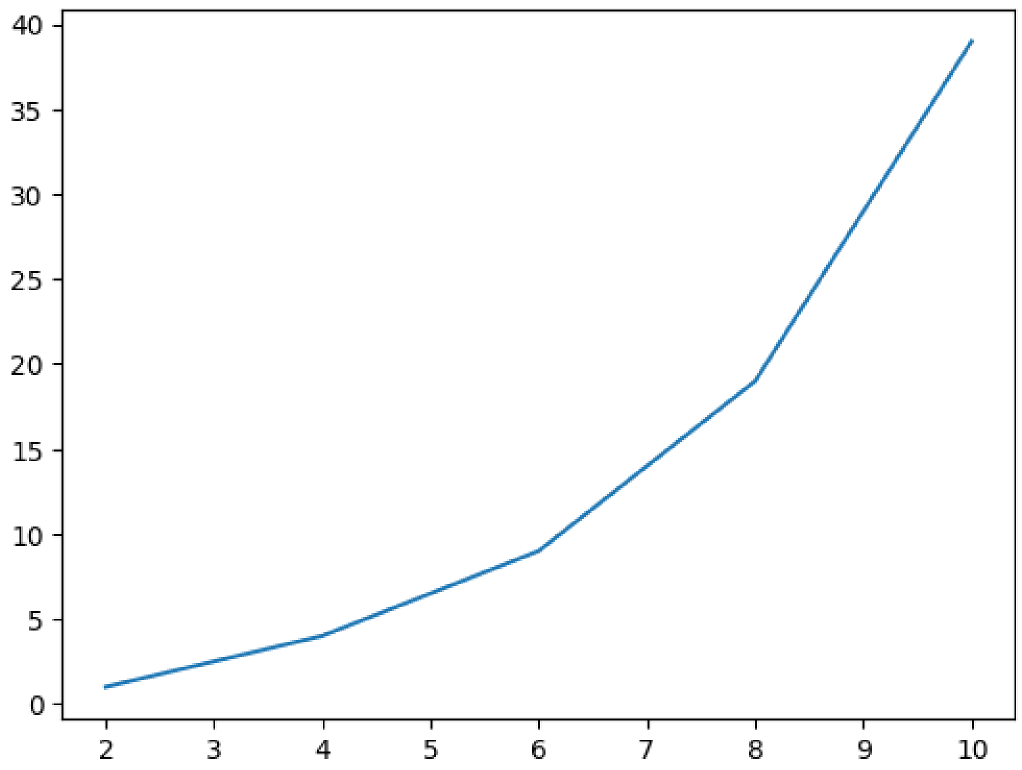
3.2 Chart Types and Usage

3.2.1 Bar Chart

Bar chart is a frequency chart for qualitative variable (or categorical variable). Bar chart can be used to assess the most-occurring and least-occurring categories within a dataset.

[58]:
```
"""
The function subplots can be used to create a figure and a set of subplots. In␣
  ↪our previous example, we
called the function without parameters which creates a figure with a single␣
  ↪included axes.
"""
import matplotlib.pyplot as plt
# Data for plotting
X = [2, 4, 6, 8, 10]
Y = [1, 4, 9, 19, 39]
fig, ax = plt.subplots()
ax.plot(X, Y)
```

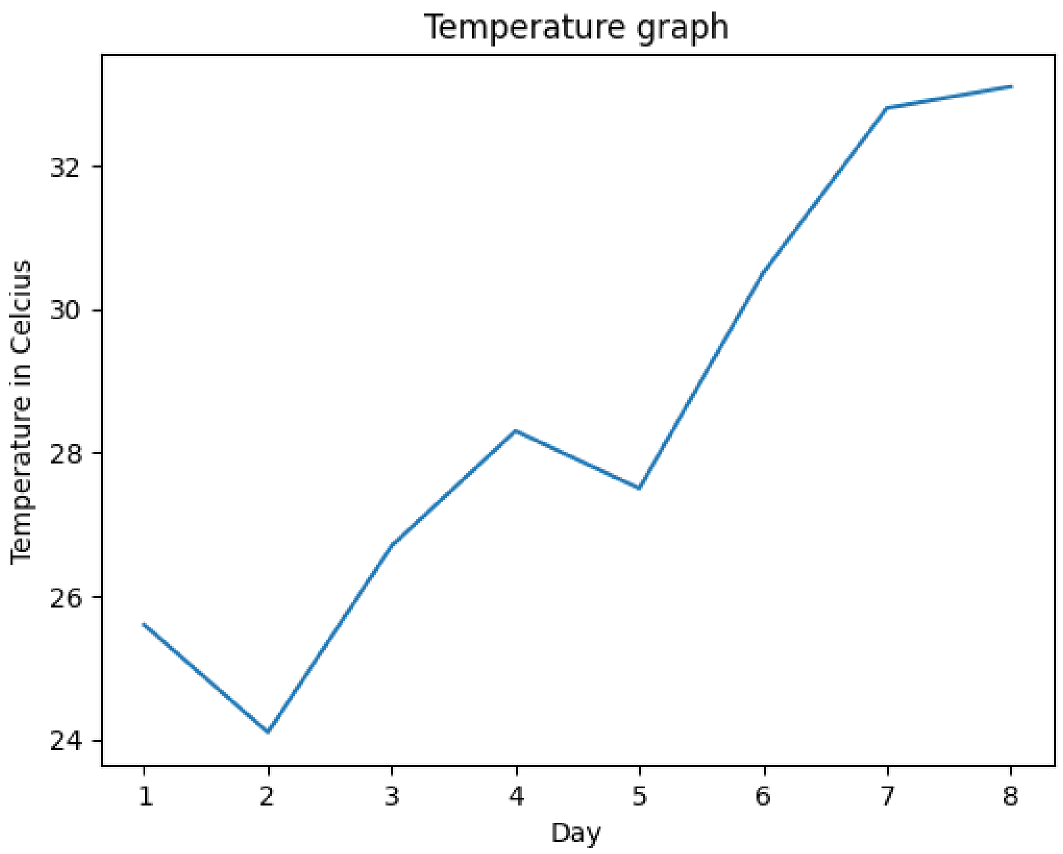
[58]:
```
[<matplotlib.lines.Line2D at 0x22b06cc1160>]
```



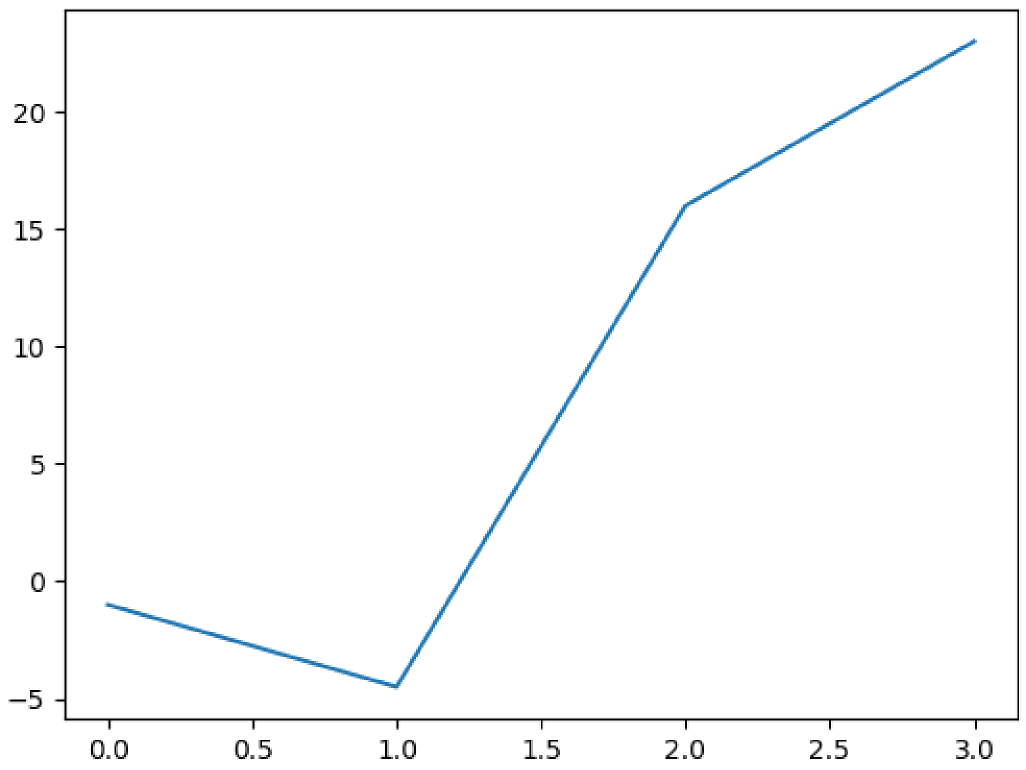
3.3 LABELS ON AXES

We can improve the appearance of our graph by adding labels to the axes. We also want to give our plot a headline or let us call it a title to stay in the terminology of Matplotlib. We can accomplish this by using the set method of the axis object ax :

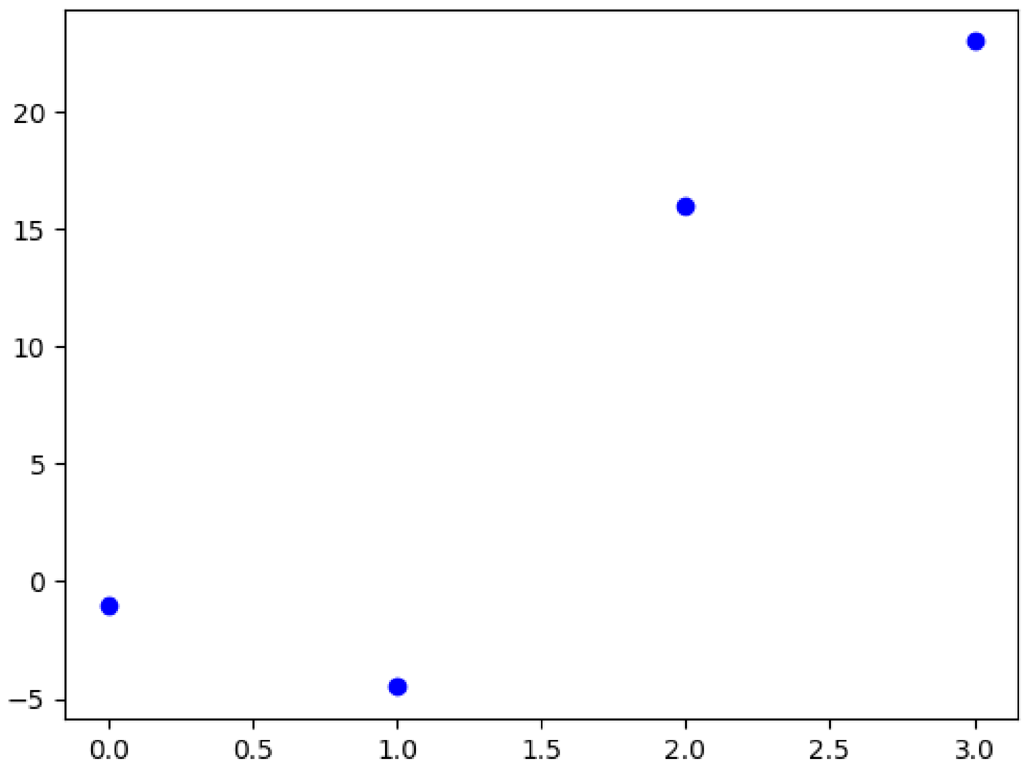
[61]:
```
import matplotlib.pyplot as plt
days = list(range(1,9))
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
fig, ax = plt.subplots()
ax.plot(days,celsius_values )
ax.set(xlabel="Day",
       ylabel="Temperature in Celcius",
      title="Temperature graph")
plt.show()
```



[6]:
```
plt.plot([-1,-4.5,16,23])
plt.show()
```



[7]:
```
plt.plot([-1,-4.5,16,23], "ob")
plt.show()
```



4 THE FORMAT PARAMETER OF PYPLOT.PLOT

THE FORMAT PARAMETER OF PYPLOT.PLOT We have used “ob” in our previous example as the format parameter. It consists of two characters. The first one defines the line style or the dicrete value style, i.e. the markers, while the second one chooses a colour for the graph.

The following color abbreviations are supported:

===================

character color

===================

‘b’ blue

‘g’ green

‘r’ red

‘c’ cyan

‘m’ magenta

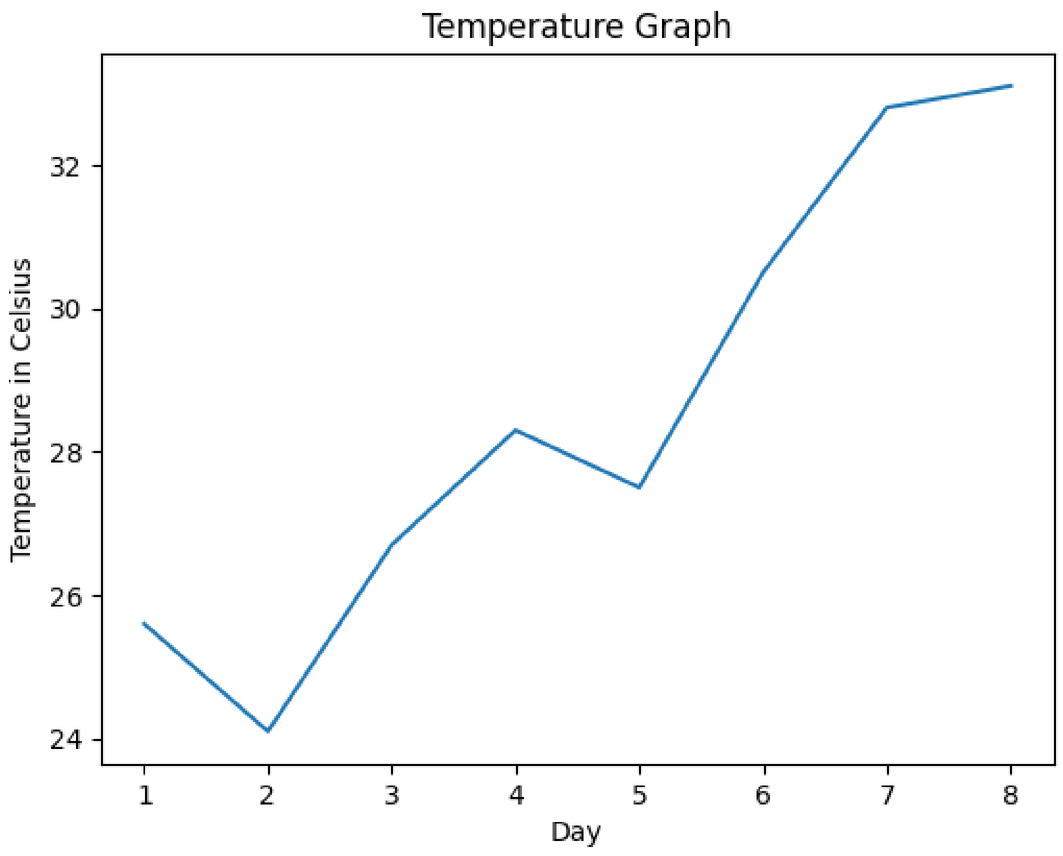
‘y’ yellow

‘k’ black

‘w’ white

===================

```
[19]: import matplotlib.pyplot as plt
days = range(1, 9)
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
fig, ax = plt.subplots()
ax.plot(days, celsius_values)
ax.set(xlabel='Day',
    ylabel='Temperature in Celsius',
    title='Temperature Graph')
plt.show()
```



4.1 THE PLOT FUNCTION

The plot function is needed to plot a figure or better the figures, because there may be more than one. When plot is run in ipython with its pylab mode, it displays all figures and returns to the ipython prompt. When we run it in non-interactive mode, - which is the case, when we run a Python program - it displays all figures and blocks until the figures have been closed.

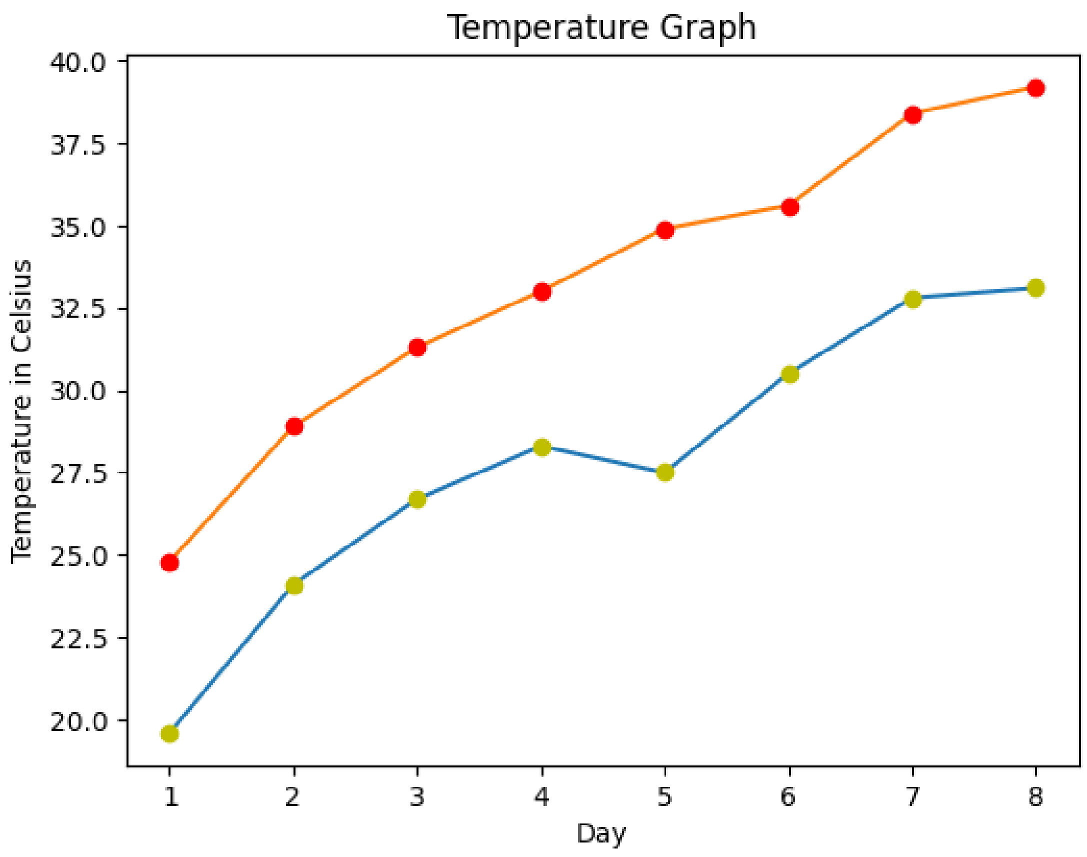
We can specify an arbitrary number of x, y, fmt groups in a plot function. We will extend our previous temperature example to demonstrate this. We provide two lists with temperature values, one for the minimum and one for the maximum values:

[31]:
```python
import matplotlib.pyplot as plt
days = list(range(1,9))
celsius_min = [19.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
celsius_max = [24.8, 28.9, 31.3, 33.0, 34.9, 35.6, 38.4, 39.2]

fig, ax = plt.subplots()

ax.set(xlabel='Day',
    ylabel='Temperature in Celsius',
    title='Temperature Graph')

ax.plot(days, celsius_min,
    days, celsius_min, "oy",
    days, celsius_max,
    days, celsius_max, "or")
plt.show()
```

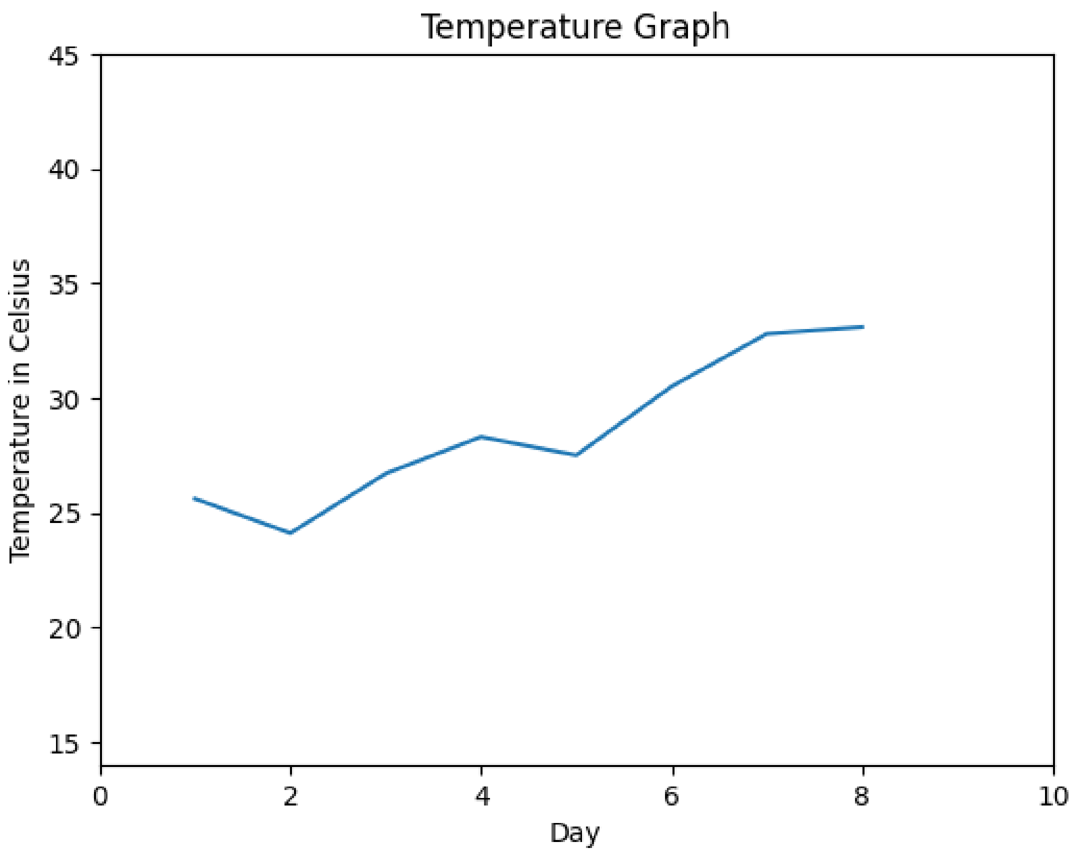


4.2 CHECKING AND DEFINING THE RANGE OF AXES

We can also check and define the range of the axes with the function axis. If you call it without arguments it returns the current axis limits:

[63]:
```
import matplotlib.pyplot as plt
days = list(range(1, 9))
celsius_values = [25.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]
fig, ax = plt.subplots()
ax.plot(days, celsius_values)
ax.set(xlabel='Day',
ylabel='Temperature in Celsius',
title='Temperature Graph')
print("The current limits for the axes are:")
print(ax.axis())
print("We set the axes to the following values:")
xmin, xmax, ymin, ymax = 0, 10, 14, 45
print(xmin, xmax, ymin, ymax)
ax.axis([xmin, xmax, ymin, ymax])
plt.show()
```

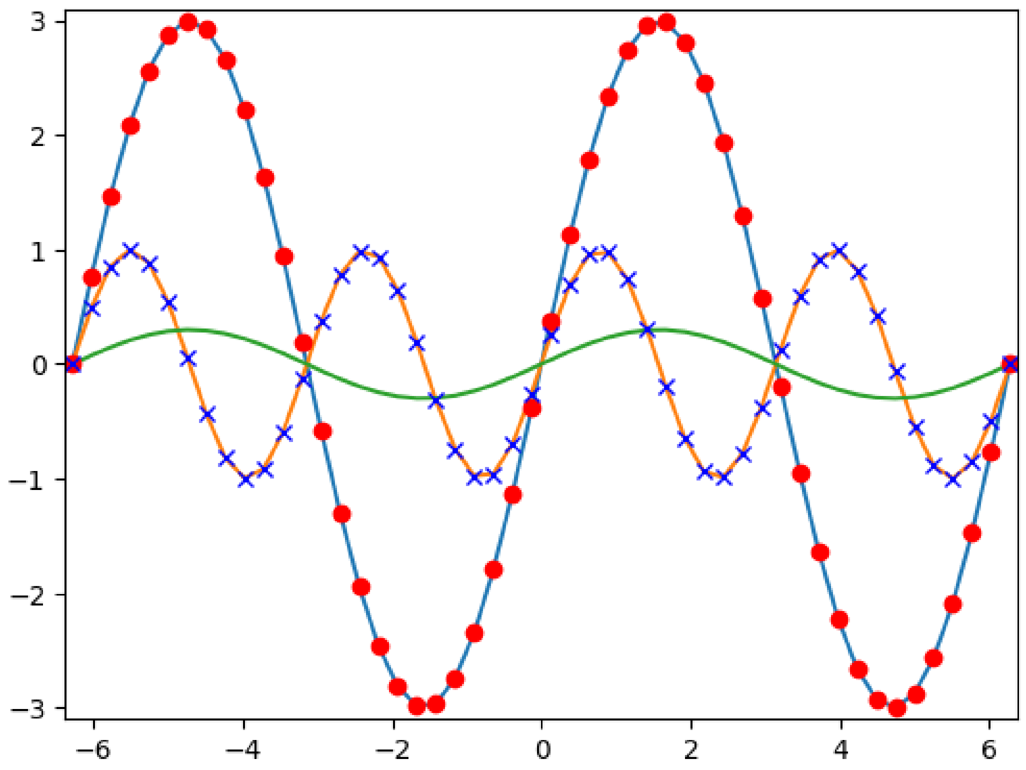
```
The current limits for the axes are:
(0.6499999999999999, 8.35, 23.650000000000002, 33.550000000000004)
We set the axes to the following values:
0 10 14 45
```



4.3 LINSPACE” TO DEFINE X VALUES

We will use the Numpy function linspace in the following example. linspace can be used to create evenly spaced numbers over a specified interval.

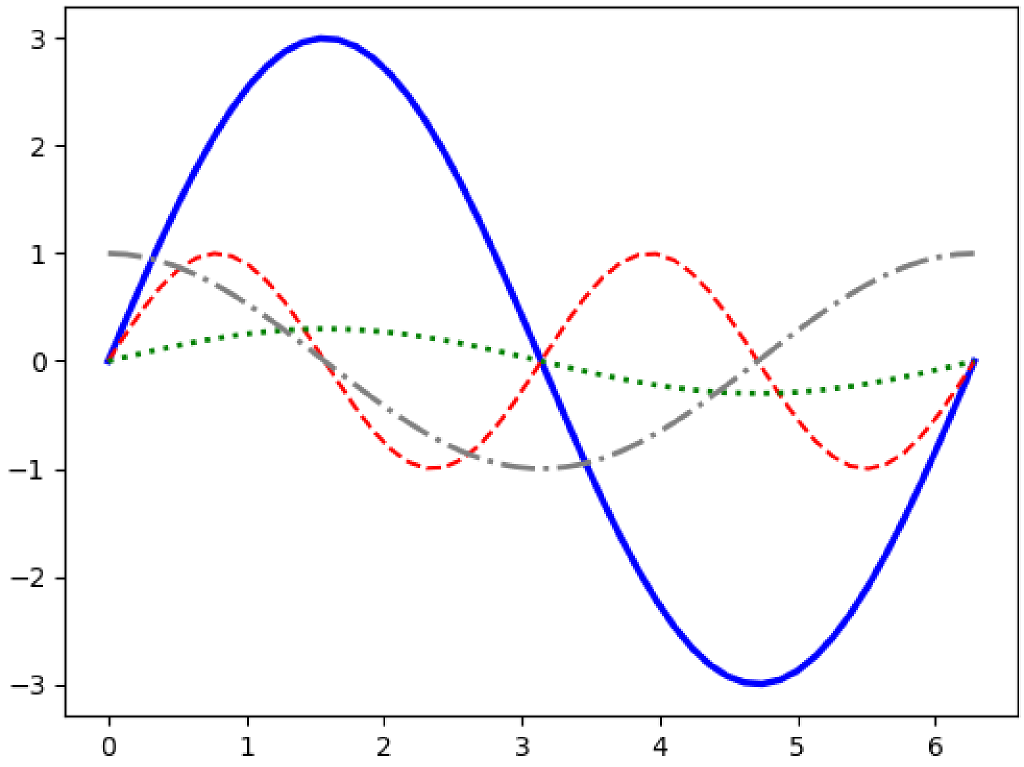
[66]:
```python
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-2 * np.pi, 2 * np.pi, 50, endpoint=True)
F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
F3 = 0.3 * np.sin(X)
fig, ax = plt.subplots()
startx, endx = -2 * np.pi - 0.1, 2*np.pi + 0.1
starty, endy = -3.1, 3.1
ax.axis([startx, endx, starty, endy])
ax.plot(X, F1, X, F2, X, F3)
# discrete points:
ax.plot(X, F1, 'ro', X, F2, 'bx')
plt.show()
```



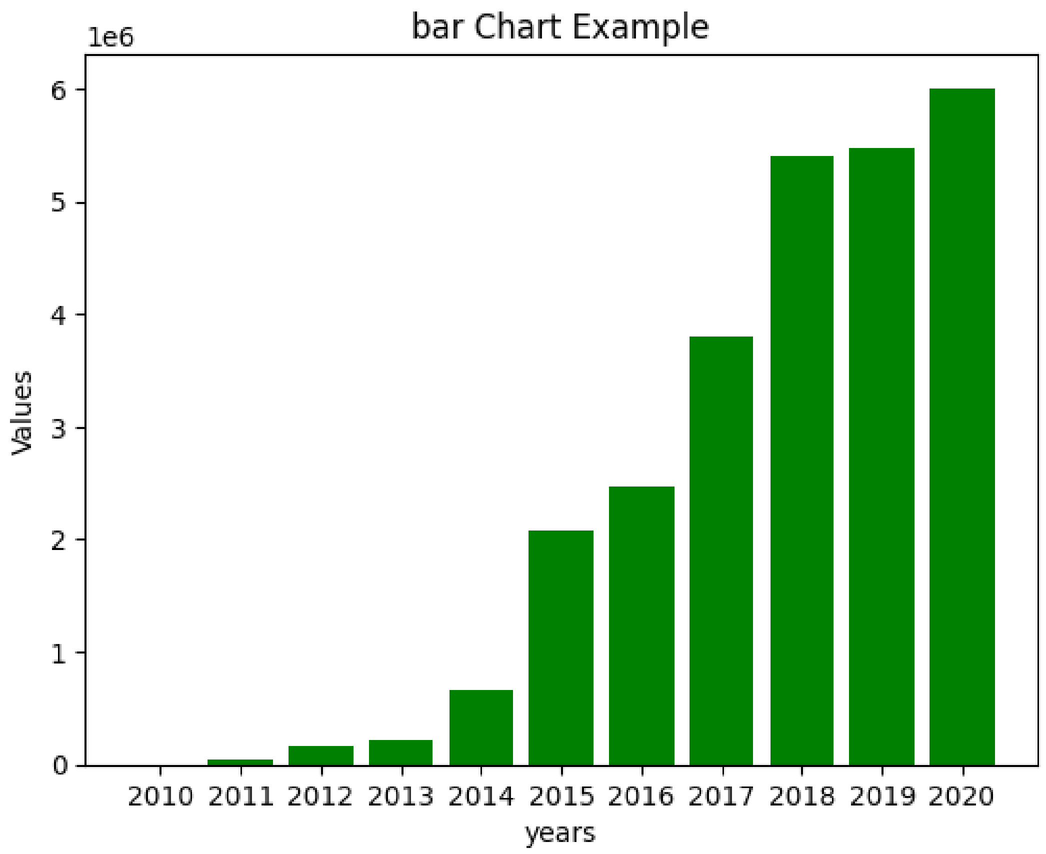
4.4 CHANGING THE LINE STYLE

The linestyle of a plot can be influenced with the linestyle or ls parameter of the plot function. It can be set to one of the following values: '-', '–', '-.', ':', 'None', ' ',' We can use linewidth to set the width of a line as the name implies.

[67]:
```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(0, 2 * np.pi, 50, endpoint=True)
F1 = 3 * np.sin(X)
F2 = np.sin(2*X)
F3 = 0.3 * np.sin(X)
F4 = np.cos(X)
fig, ax = plt.subplots()
ax.plot(X, F1, color="blue", linewidth=2.5, linestyle="-")
ax.plot(X, F2, color="red", linewidth=1.5, linestyle="--")
ax.plot(X, F3, color="green", linewidth=2, linestyle=":")
ax.plot(X, F4, color="grey", linewidth=2, linestyle="-.")
plt.show()
```

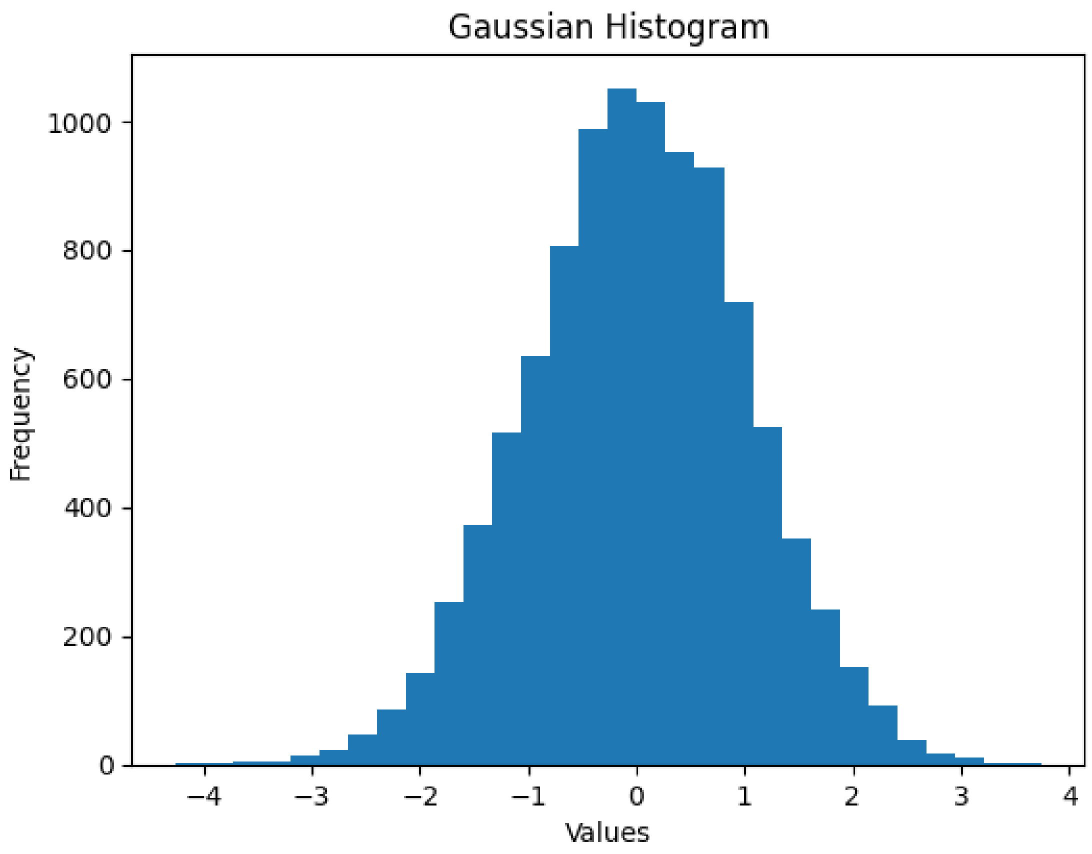


[35]:
```
import matplotlib.pyplot as plt
import numpy as np
years = [str(year) for year in range(2010, 2021)]
visitors = (1241, 50927, 162242, 222093, 665004, 2071987, 2460407, 3799215,␣
  ↪5399000, 5474016, 6003672)
plt.bar(years, visitors, color = "green")
plt.xlabel("years")
plt.ylabel("Values")
plt.title("Bar Chart Example")
plt.plot()
plt.show()
```



[44]:
```
import matplotlib.pyplot as plt
import numpy as np
gaussian_numbers = np.random.normal(size=10000)
gaussian_numbers
plt.hist(gaussian_numbers, bins=30)
plt.title("Gaussian Histogram")
plt.xlabel("Values")
plt.ylabel("Frequency")
```

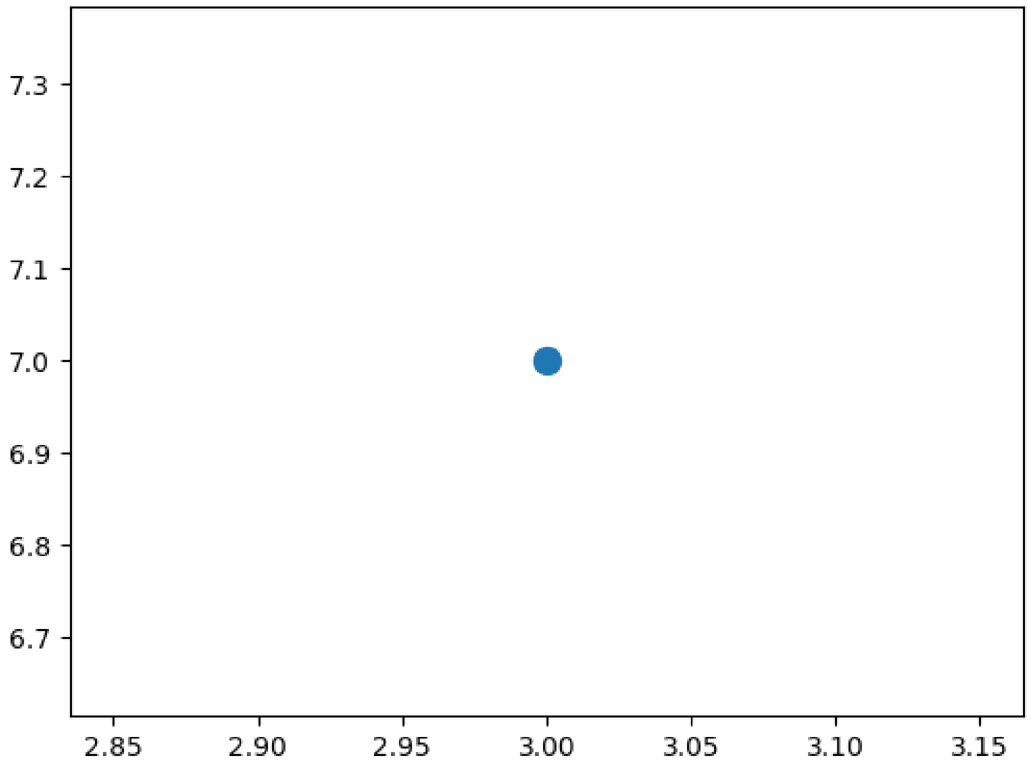
[44]:
```
Text(0, 0.5, 'Frequency')
```



4.5 SCATTER PLOTS IN MATPLOTLIB

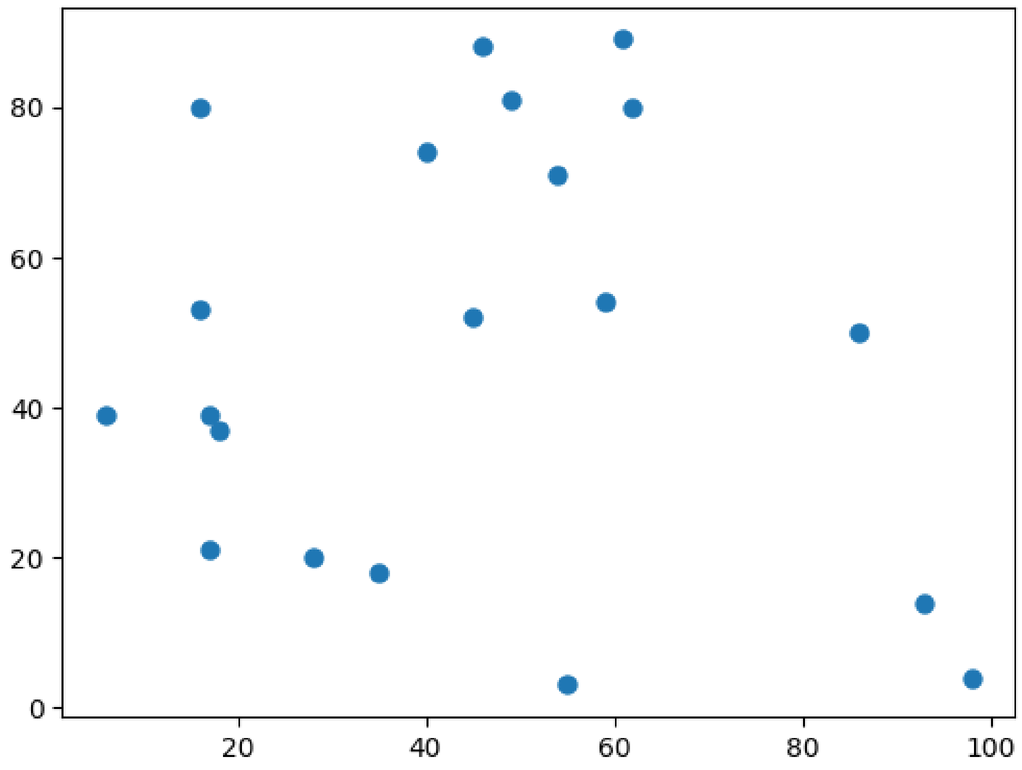
[69]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.scatter(3, 7, s=100)
```

[69]:
```
<matplotlib.collections.PathCollection at 0x22b04eefce0>
```

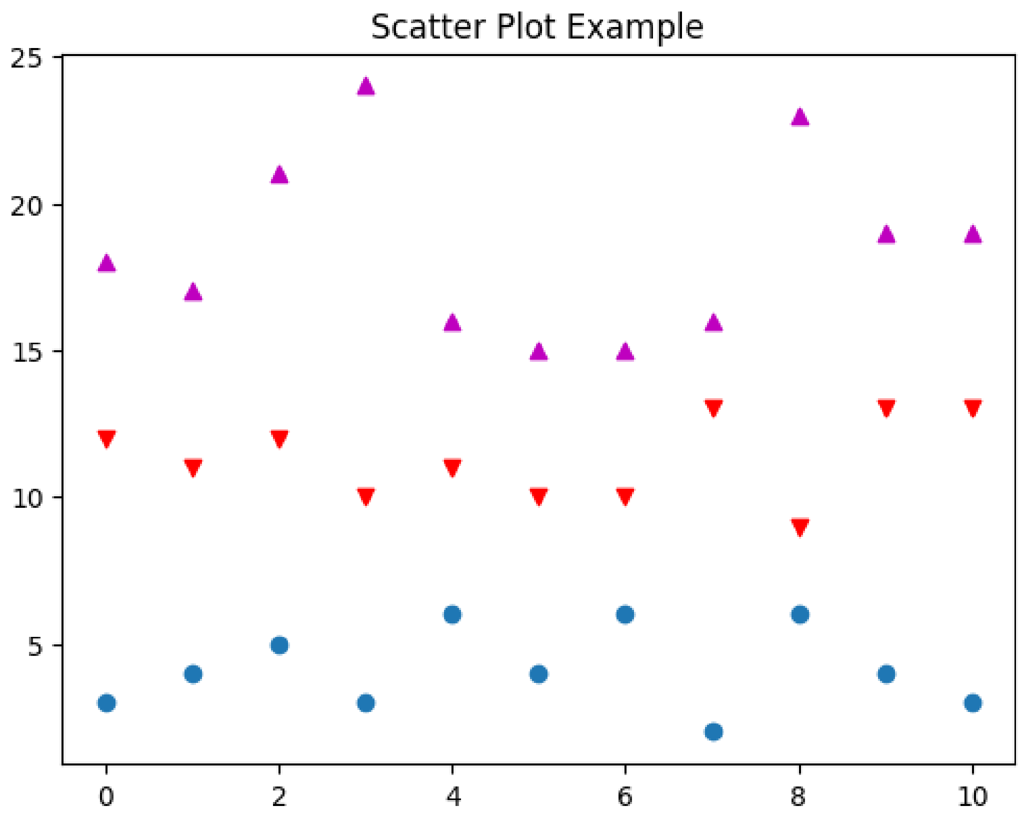


```
[71]: import matplotlib.pyplot as plt
      import numpy as np
      X = np.random.randint(0, 100, (20,))
      Y = np.random.randint(0, 100, (20,))
      fig, ax = plt.subplots()
      ax.scatter(X, Y, s=42)
```

```
[71]: <matplotlib.collections.PathCollection at 0x22b047c5490>
```



[49]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0,11)
y1 = np.random.randint(2,7,(11,))
y2 = np.random.randint(9,14,(11,))
y3 = np.random.randint(15,25,(11,))
plt.scatter(x, y1)
plt.scatter(x,y2, marker='v', color='r')
plt.scatter(x, y3, marker='^', color='m')
plt.title('Scatter Plot Example')
plt.show()
```

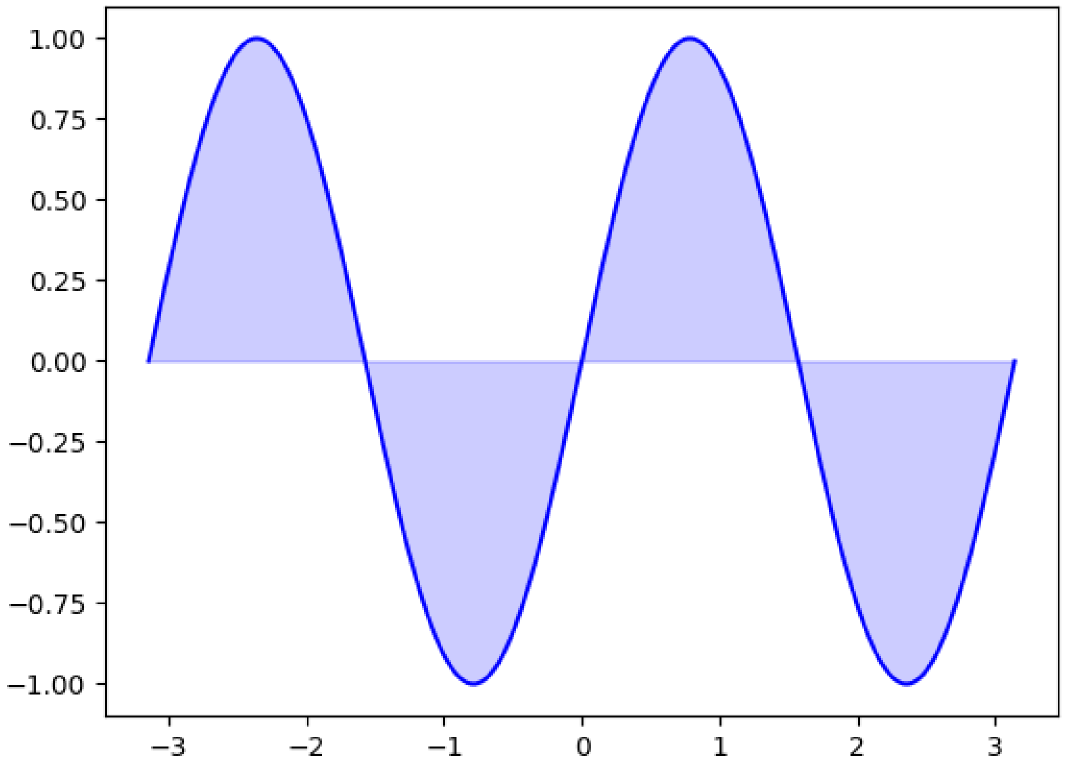


4.6 Shading regions with fill_etween()

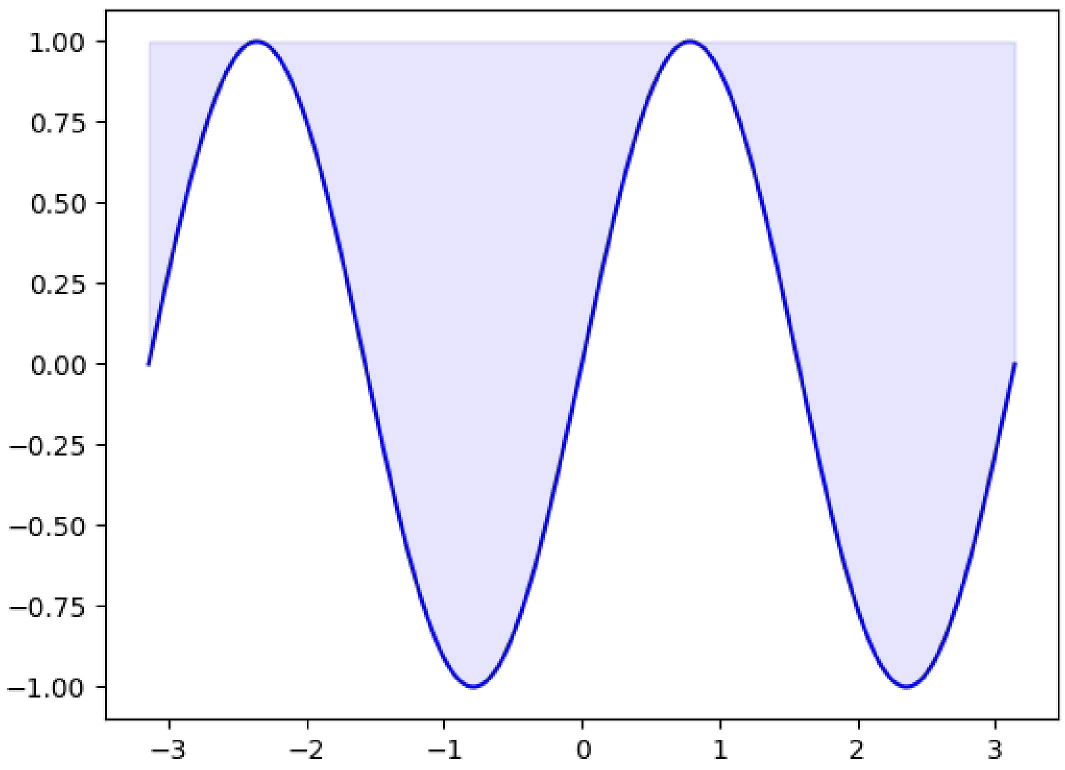
It is possible to shade or colorize regions between two curves. We are filling the region between the X axis and the graph of sin(2*X) in the following example.

The general syntax of fill_between: fill_between(x, y1, y2=0, where=None, interpolate=False, **kwargs)

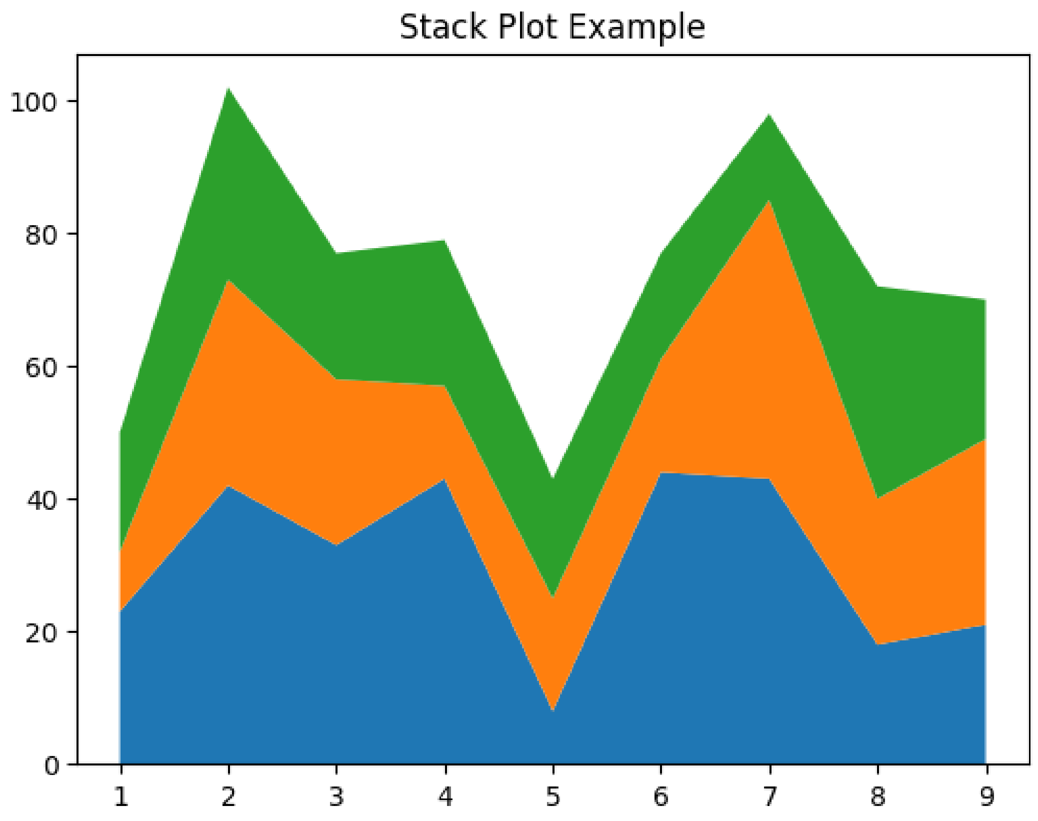
[73]:
```python
import numpy as np
import matplotlib.pyplot as plt
n = 256
X = np.linspace(-np.pi,np.pi,n,endpoint=True)
Y = np.sin(2*X)
fig, ax = plt.subplots()
ax.plot (X, Y, color='blue', alpha=1.0)
ax.fill_between(X, 0, Y, color='blue', alpha=.2)
plt.show()
```



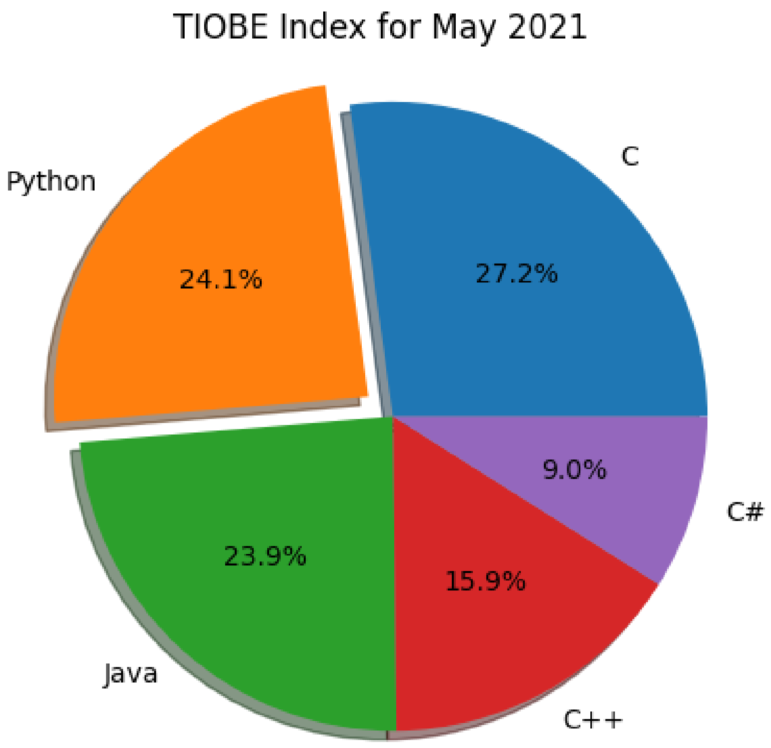
[74]:
```
import numpy as np
import matplotlib.pyplot as plt
n = 256
X = np.linspace(-np.pi,np.pi,n,endpoint=True)
Y = np.sin(2*X)
fig, ax = plt.subplots()
ax.plot (X, Y, color='blue', alpha=1.00)
ax.fill_between(X, Y, 1, color='blue', alpha=.1)
plt.show()
```



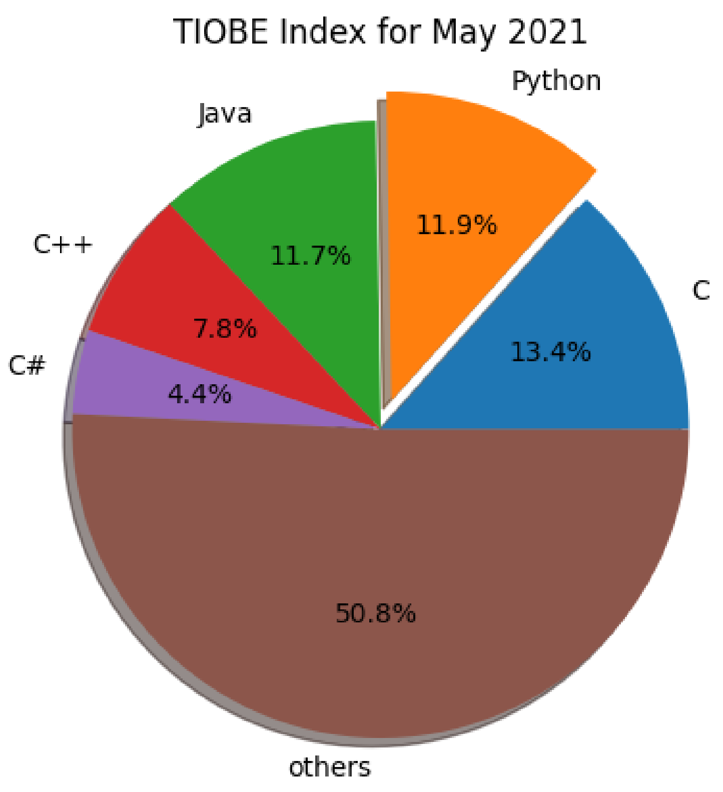
[50]:
```
import matplotlib.pyplot as plt
idxes = [ 1, 2, 3, 4, 5, 6, 7, 8, 9]
y1 = [23, 42, 33, 43, 8, 44, 43, 18, 21]
y2 = [9, 31, 25, 14, 17, 17, 42, 22, 28]
y3 = [18, 29, 19, 22, 18, 16, 13, 32, 21]
plt.stackplot(idxes,
y1, y2, y3)
plt.title('Stack Plot Example')
plt.show()
```



```
[54]: import matplotlib.pyplot as plt
      # Pie chart, where the slices will be ordered and plotted counter-clockwise:
      labels = 'C', 'Python', 'Java', 'C++', 'C#'
      sizes = [13.38, 11.87, 11.74, 7.81, 4.41]
      explode = (0, 0.1, 0, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')
      fig1, ax1 = plt.subplots()
      ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
      shadow=True, startangle=0)
      ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
      plt.title('TIOBE Index for May 2021')
      plt.show()
```



[56]:
```
# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'C', 'Python', 'Java', 'C++', 'C#', 'others'
sizes = [13.38, 11.87, 11.74, 7.81, 4.41]
sizes.append(100 - sum(sizes))
explode = (0, 0.1, 0, 0, 0, 0) # only "explode" the 2nd slice(i.e. 'Hogs')
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',shadow=True,␣
 ↪startangle=0)
ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('TIOBE Index for May 2021')
plt.show()
```

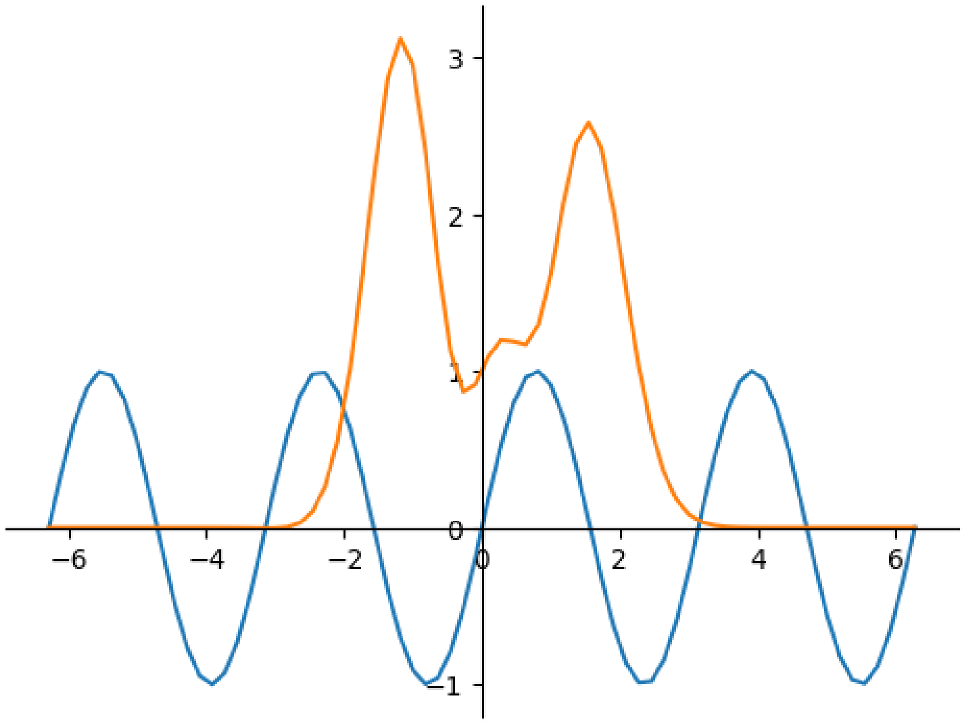


4.7 Spine and Ticks

4.7.1 MOVING THE BORDER LINES AND POLISHING UP THE AXES NOTATIONS

Spines in matplotlib are the lines connecting the axis tick marks and noting the boundaries of the data area.

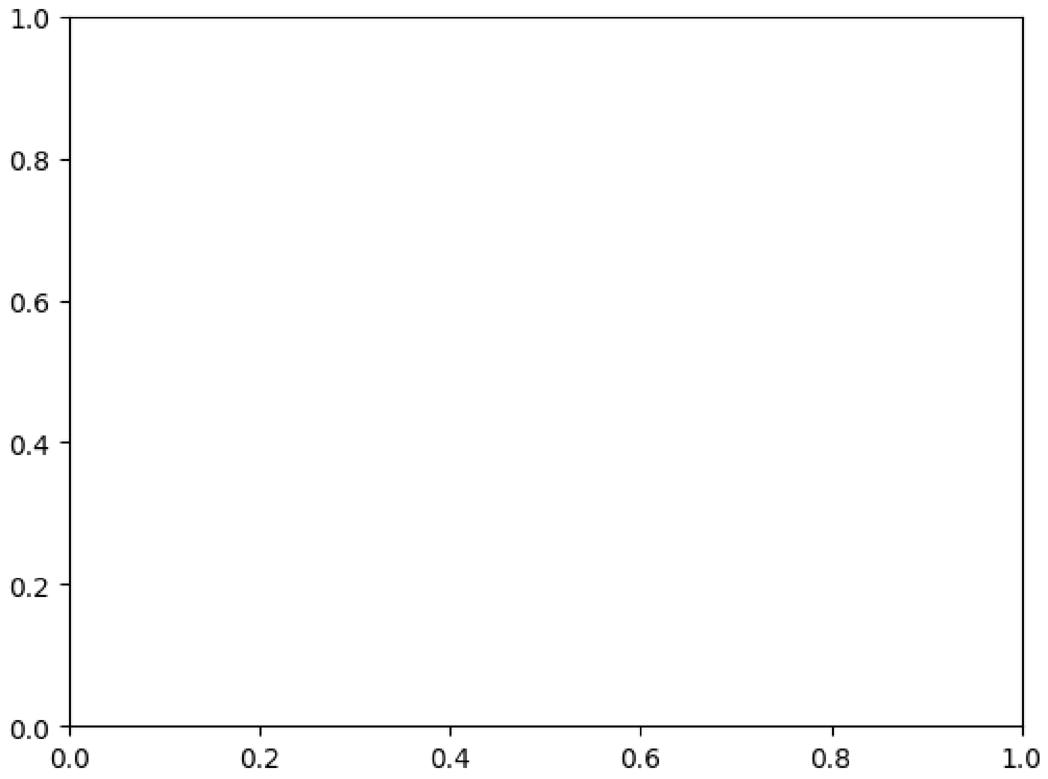
[75]:
```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-2 * np.pi, 2 * np.pi, 70, endpoint=True)
F1 = np.sin(2* X)
F2 = (2*X**5 + 4*X**4 - 4.8*X**3 + 1.2*X**2 + X + 1)*np.exp(-X**2)
fig, ax = plt.subplots()
# making the top and right spine invisible:
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
# moving bottom spine up to y=0 position:
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
# moving left spine to the right to position x == 0:
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
ax.plot(X, F1, X, F2)
plt.show()
```



4.7.2 CUSTOMIZING TICKS

[77]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
xticks = ax.get_xticks()
xticklabels = ax.get_xticklabels()
print(xticks, xticklabels)
for i in range(6):
    print(xticklabels[i])
yticks = ax.get_yticks()
print(yticks)
```

```
[0.  0.2 0.4 0.6 0.8 1. ] [Text(0.0, 0, '0.0'), Text(0.2, 0, '0.2'), Text(0.4,
0, '0.4'), Text(0.6000000000000001, 0, '0.6'), Text(0.8, 0, '0.8'), Text(1.0, 0,
'1.0')]
Text(0.0, 0, '0.0')
Text(0.2, 0, '0.2')
Text(0.4, 0, '0.4')
Text(0.6000000000000001, 0, '0.6')
Text(0.8, 0, '0.8')
Text(1.0, 0, '1.0')
[0.  0.2 0.4 0.6 0.8 1. ]
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0